IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

PS-ECBS: A New Algorithm for Multi-Agent Path Finding with Optimal Task Assignment

Cheng Zhang, Shixin Liu

Abstract— **Multi-agent path finding (MAPF) problem in warehouse automation consists of optimal task assignment and path planning, where small runtime is necessary. In this paper, we present a new MAPF algorithm related to dynamic start and end positions of the robots, called Position-Selection Enhanced Conflict-Based Search (PS-ECBS). Conflict-Based Search (CBS) is a well-known framework that has been used to find collision-free paths for a given fixed task assignment, while ECBS is a bounded-suboptimal variant of CBS that uses focal search to speed up CBS. The mixed integer linear programming (MILP) is introduced to formulate the dynamic model for optimal task assignment, and the successful combination of MILP and ECBS results in PS-ECBS algorithm. The solving process of the PS-ECBS consists of multiple iterations, and in each iteration an additional constraint is added to modify the model. In the computational experiment, the processes of picking up and putting back shelves in the warehouse could occur at the same time by PS-ECBS. We also analyzed the iterative principle of PS-ECBS, and compared its performance with that of ECBS-TA. The computational results demonstrate that PS-ECBS runs significantly faster and has an obvious advantage in jointly optimizing task assignment and path planning for large-scale warehouse.**

I. Introduction

Robotic mobile picking systems (RMPS) have high operation efficiency. Its central idea is to use a mobile robot to carry the movable shelf containing ordered items to the picking station, and after the goods are picked by the staff, the robot will return the shelf to an empty position in the storage area [1], [2], [3]. The path planning of RMPS is a kind of multi-agent path finding (MAPF) problem, which is to find collision-free paths from initial locations to destinations for multiple agents representing robots or vehicles in a known environment [4], [5], [6]. MAPF has been applied in many fields, including warehouse logistics [7], [8], [9], [10], [11], driverless operation [12], autonomous vehicles [13], digital entertainment [14]. In 2012, Amazon first applied the MAPF of Kiva mobile robot to intelligent logistics [15], which greatly improved the order picking efficiency, and then the research and application on MAPF in warehouse logistics flourished.

It is worth noting that the high-efficiency algorithm is the crucial factor in MAPF. There are many classical algorithms in MAPF, including Dijkstra [16], A*[17], M* [18], integer linear programming [19] and CBS [20], [21]. The CBS algorithm, being an advanced MAPF algorithm, has gradually attracted researchers' attention in recent years. The CBS is a two-level search algorithm for solving MAPF optimally. It plans a path for each agent independently and then resolves collisions between agents by branching. Each branch is a new candidate plan wherein one agent or the other is forced to find a new path that avoids the chosen collision. In order to optimize CBS, many CBS-based algorithms have been developed by extending CBS, including Improved CBS (ICBS) [22], Enhanced CBS (ECBS) [23], Explicit Estimation CBS (EECBS) [24], CBS with Continuous-time (CCBS) [25], *k*-Robust CBS (*k*R-CBS) [26], ECBS with Region Heuristics (RH-ECBS) [27], ECBS with Task Assignment (ECBS-TA) [28].

These CBS variants primarily focus on guaranteeing theoretical optimality of solutions (e.g., ICBS), improving the solving speed of path planning (e.g., ECBS, EECBS), and enhancing the overall efficiency of the planning process (e.g., CCBS, kR-CBS, RH-ECBS). However, it should be noted that most MAPF research predominantly targets the labeled case, where the goal for each agent is preassigned. ECBS-TA is a MAPF algorithm capable of optimal goal selection, yet there remains a notable lack of research reports on its application to realistic scenarios, such as path planning of robots in warehouse.

In the traditional picking system, the task assignment and path planning are two problems to solve independently in dense environments. However, MAPF problem is composed of optimal task assignment and path planning in warehouse automation, both of which are tightly coupled, and certain task assignments may result in fewer collisions during path planning. Mixed-Integer Linear Programming (MILP) excels at handling discrete decision-making (e.g., selecting start and end points) and global optimization, which can effectively model the system's operational constraints and objectives within a mathematical framework [29]. Recent research has explored the integration of MILP with heuristic methods, aiming to combine MILP's optimization capabilities with the speed and adaptability of heuristic searches [30]. In this paper, we propose a new algorithm for solving MAPF problems related to dynamic start and end positions of the robots in the warehouse, called Position-Selection ECBS (PS-ECBS). Our PS-ECBS approach combines MILP-structured iterative constraints with the ECBS framework for optimal position selection in warehouse scenarios, where the robot needs to decide which shelf to pick up or which empty position to return the shelf to for optimal task assignments. The PS-ECBS algorithm is a two-stage iterative algorithm. In the first stage, the algorithm relaxes the robot's path conflict constraints and assigns starting points or destination end points to robots using the MILP model. In the second stage, the PS-ECBS calls the ECBS to plan the path for robots. If the objective function

*This work is supported by National Natural Science Foundation of China (62073069) and LiaoNing Revitalization Talents Program (XLYC2002041).

Cheng Zhang and Shixin Liu are with the College of Information Science and Engineering, Northeastern University, Shenyang, 110819, China. (e-mail: 2010304@stu.neu.edu.cn, sxliu@mail.neu.edu.cn).

value of the second stage is larger than the one of the first stage, a new constraint is added to the MILP to avoid path conflicts arising from the current assignment, and a new iterative process is started until the algorithm converges.

We elaborate on the iteration principle and convergence condition of the PS-ECBS algorithm, and compare its performance with that of ECBS-TA, which is a method to jointly optimize for task assignment and path planning in the ECBS framework. The results demonstrate that PS-ECBS runs significantly faster than ECBS-TA and shows the obvious superiority in joint optimization for large-scale warehouse.

II. Description of Warehouse Background

The warehouse layout is shown in Fig. 1, which consists of the coordinates of shelves and picking stations, as well as the moving paths of robots. There may be one or more kinds of goods on each shelf. According to a personalized order batch, a task requires robots to pick up some particular shelves, bring them to the specified picking stations, and then return the shelves to empty positions in the storage area. The main idea is to improve the efficiency of warehouse logistics through MAPF as follows:

- In picking up task, the robots should initially select target shelves for retrieval, while avoiding collisions between robots both during transit and at goal positions.
- In putting back task, the robots should select empty positions to put shelves back into, while maintaining collision-free paths. We compare the performance between PS-ECBS and ECBS-TA in this scenario.
- Both processes mentioned above should occur simultaneously in the warehouse, and the joint optimization of multi-robot task assignment and path planning should be achieved, where small runtime is necessary.

III. Picking up Task in Warehouse

A. MILP Model in Picking up Task

In the picking up process, the robot should carry the selected shelves $i(i = 1, \dots, N)$ at (X_i, Y_i) and move toward the picking station at (X_0^p, Y_0^p) along the dotted line in Fig. 1. The length c_i of the shortest path from shelf i to picking station is the Manhattan distance, which can be initially calculated by (1), where (X_i, Y_i) is the coordinate of the ith shelf, (X_0^p, Y_0^p) is the coordinate of the pth picking station $(p = 1,2, \cdots)$.

$$c_i = |X_i - X_0^p| + |Y_i - Y_0^p| \tag{1}$$

According to Fig. 1, in area 1, area 2 and area 3, the shelves on the left should be transported to the picking station in the same area, and the shelves in the picking station should be returned to the left empty positions in the same area. If the total number of shelves in one row is U, in area1, $0 < i \le 3U$, $p = 1$. In area 2, $3U < i \le N - 3U$, $p = \lfloor (i - U)/2U \rfloor + 1$. In area 3, if $N - 3U < i \le N - U$, $p = \lfloor (i - U)/2U \rfloor + 1$, if $N - U < i \le N$, $p = \lfloor (i - U)/2U \rfloor$.

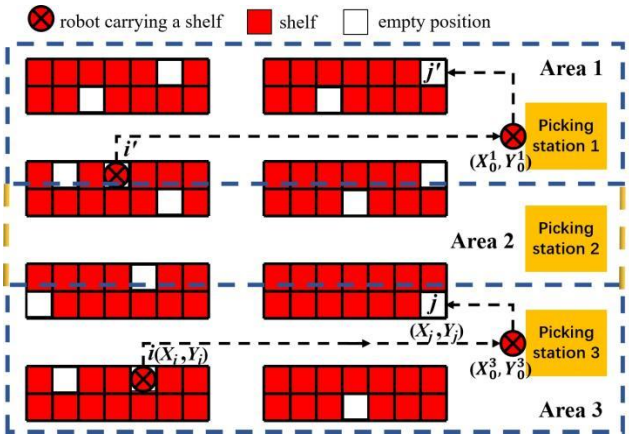


Fig. 1 Schematic diagram of RMPF path planning in the warehouse. Any robot needs to pick up the shelf at position i to the picking station first, and then put it back to an empty position j in the storage area. Similarly, it can also transport the shelf at position i' to the picking station, and then store it at position j'.

Suppose the warehouse receives a batch of orders. Where, the demand of commodity $l(l = 1, \dots, L)$ is d_l, the quantity of commodity l in shelf i is a_{il}, and the distance from shelf i to the picking station is c_i. Then, the shelf selection problem in stage 1 is: how to select shelves, so that the commodities on the selected shelves meet the order demands, and the total distance to the picking point is minimum. Define decision variable x_i, if shelf i is selected, then $x_i = 1$, otherwise $x_i = 0$. The MILP model 1 for the position selection stage is formulated as (2)~(5).

$$\min \quad z \tag{2}$$

$$z \ge \sum_{i=1}^{N} c_i x_i \tag{3}$$

$$\sum_{i=1}^{N} a_{il} x_i \ge d_l, l = 1, \dots, L \tag{4}$$

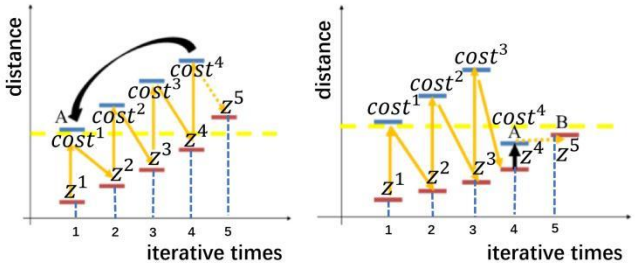
$$x_i \in \{0, 1\}, i = 1, \dots, N \tag{5}$$

Where, in the model the objective is to minimize z as shown in (2). Because the real distance includes the distance to avoid collision between robots, which is greater than or equal to the ideal distance shown in (3). Constraint (4) is to meet the order demands. Constraint (5) is variable definition.

After determining all the selected shelves by the above model, ECBS algorithm could be used for path planning of the robots and cost calculating. However, the best solution may not be the result calculated through the MILP. In other words, if the solution obtained using MILP model is input into ECBS algorithm, the cost may not be the minimum. This is because when objective function $z = \sum_{i=1}^{N} c_i x_i$ is the ideal distance, in which the path conflicts in the process of transporting shelves by robots are not considered, and the cost by ECBS reflects the sum of the path costs for all robots in reality. Hence, there may be differences between ideal distance z and actual cost.

B. Iterative Algorithm for the Shelf Selection Stage

Let z be the ideal distance obtained by solving model (2)~(5), and $cost$ be the actual distance obtained by ECBS



(a) cost increasing all time (b) cost not increasing all the time

Fig. 2 Schematic diagram of iterative principle, where the red line represents z obtained by solving MILP, and the blue line represents the cost obtained by ECBS.

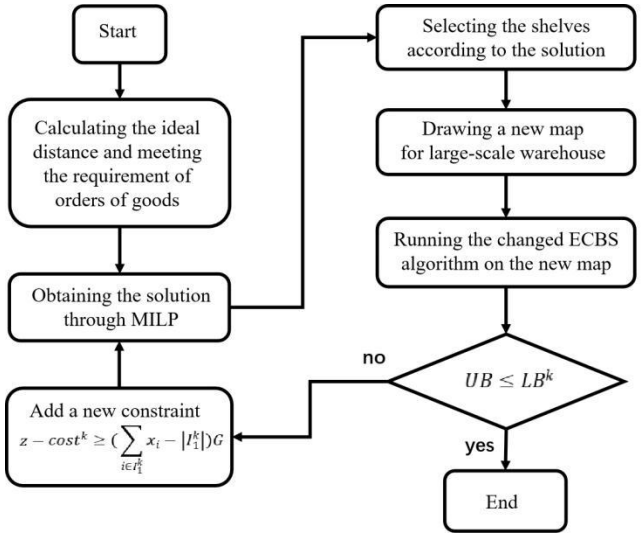


Fig. 3 Schematic diagram of PS-ECBS process for picking up task.

based on the model solution. If cost is much larger than the ideal distance $z = \sum_{i=1}^{N} c_i x_i$, an additional constraint should be added to the model (2)~(5) to obtain another improved solution, and so on. Therefore, the solving process of the algorithm is an iterative process.

Let z^k be the ideal distance obtained in the kth iteration by solving model (2)~(5), and the corresponding solution is $x^k = (x_1^k, x_2^k, \cdots, x_N^k)$, where subscripts of variable x successively increase from left to right and from top to bottom according to Fig. 1. According to the values of the decision variables, we divide the decision variables into two sets, one is $\{x_i^k | x_i^k = 0, i \in I_0^k\}$, and the other is $\{x_i^k | x_i^k = 1, i \in I_1^k\}$. Let $cost^k$ be the actual distance obtained by ECBS based on the solution x^k. Therefore, we have $z^k \leq cost^k$. In order to modify MILP model, we should add an additional constraint to get close to the convergence result in the $(k+1)$th iteration. Hence, we add a constraint (6) into the model (2)~(5) in iteration $k+1$. If there are k iterations, k constraints should be added. In constraint (6), G is a constant large enough, which ensures obtaining the actual cost if the solver finds the identical solutions. Constraint (6) is dynamically updated in each iteration.

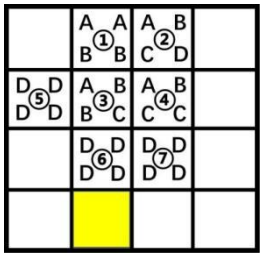


Fig. 4 Distribution of shelves and commodities on the shelves in small-scale field, where the shelves are selected by PS-ECBS based on the commodity orders.

$$z - cost^k \geq (\sum_{i \in I_1^k} x_i - |I_1^k|)G \tag{6}$$

C. Iterative Principle and Convergence Condition of PS-ECBS Algorithm

Fig. 2 shows the evolution of the objective function z obtained by solving model and *cost* obtained by ECBS based on the model solution when constraint (6) is added in each iteration. The calculation sequence of z and *cost* values is shown by the yellow arrow in Fig. 2. In the first iteration, the lower red line and upper blue line are z^1 and $cost^1$ values, respectively. In the kth iteration, we determine the upper bound (UB) of the actual distance by $UB = min\{cost^r | r = 1, \cdots, k\}$, and the lower bound of MILP is $LB^k = z^k$. If $UB - LB^k > 0$, constraint (6) should be added to the model and the algorithm goes to the next iteration until a convergence result is obtained, where $UB \leq LB^k$. In the fifth iteration (Fig. 2(a)), for example, if value of z^5 is even larger than the $UB = cost^1$, the algorithm terminates, and the best solution is x^1.

However, z value indicated by red line is increasing all the time according to the MILP principle, the cost indicated by blue line cannot increase all the time and may become even lower than the previous iterations in Fig. 2(b). Fig. 2(b) shows the situation that $UB = cost^4$, $LB^5 = z^5$, and $UB \leq z^5$ where the algorithm obtains a convergence result. The best solution is x^4. In Fig. 2 (a) and (b), A and B represent the final solution and expected last MILP solution, respectively.

Fig. 3 shows the PS-ECBS process in the warehouse, the algorithm should be automatically arranged to draw a new map for the robots in each iteration shown in Fig. 3. In warehouse, the runtime spent on each iteration should be little enough. ECBS [23] is a bounded-suboptimal variant of CBS that uses focal search on both the high and low levels. Unlike usual bounded-suboptimal searches, the focal search used on the low level of ECBS is to speed up the high-level search, instead of the low-level search itself.

In large-scale warehouses, there are multiple picking stations, and conflicts between robots at any picking station cannot be ignored. To solve this problem, the original ECBS is improved by adding Conflict Disappearing for robots at the goal, named changed ECBS in Fig. 3. The modified algorithm makes robots virtually disappear upon reaching goal, enforcing single robot occupancy at each goal position, thereby eliminating goal-position collisions.

D. Example of Picking up Task

We try to apply the PS-ECBS algorithm to the example in a small-scale field to illustrate it more clearly. As shown in Fig. 4, there are 7 shelves which are adjacent to each other, and each shelf marked 1, 2,⋯, 7 respectively has different commodities such as A, B, C, D. The yellow square indicates the picking station.

The demands of commodity order are 3A, 3B and 2C in Fig. 4. According to (1), $c_1 = 3, c_2 = 4, c_3 = 2, c_4 = 3$, the solution for the first iteration is $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$, $LB^1 = z^1 = 6$, $cost^1 = 12$. Hence, $UB = cost^1 = 12 > LB^1$, the following new constraint (7) should be added to MILP model

$$z - 12 \geq (x_1 + x_4 - 2)G \tag{7}$$

In the second iteration, the solution is $x_1 = 1$, $x_2 = 1$, $x_3 = 1$, $x_4 = 0$, $LB^2 = z^2 = 9$, $cost^2 = 21$. Hence, $UB = min\{cost^1, cost^2\} = min\{12,21\} = 12 > LB^2$, the following new constraint (8) should be added

$$z - 21 \geq (x_1 + x_2 + x_3 - 3)G \tag{8}$$

In the third iteration, the solution is $x_1 = 0, x_2 = 1, x_3 = 1$, $x_4 = 1$, $LB^3 = z^3 = 9$, $cost^3 = 18$. Hence, $UB = min\{cost^1, cost^2, cost^3\} = min\{12,21,18\} = 12 > LB^3$, the following new constraint (9) should be added

$$z - 18 \geq (x_2 + x_3 + x_4 - 3)G \tag{9}$$

In the fourth iteration, the solution is $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$, $LB^4 = z^4 = 8$, $cost^4 = 18$. Hence, $UB = min\{cost^1, cost^2, cost^3, cost^4\} = min\{12,21,18,18\} = 12 > LB^4$, the following new constraint (10) should be added

$$z - 18 \geq (x_1 + x_3 + x_4 - 3)G \tag{10}$$

In the fifth iteration, the solution is $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$, $LB^5 = z^5 = 12$, $cost^5 = 12$. Hence, $UB = min\{cost^1, cost^2, cost^3, cost^4, cost^5\} = min\{12,21,18,18,12\} = 12 \leq LB^5$, the algorithm completes its iterations, and the best solution of MILP model is obtained. Therefore, the shelves marked 1 and 4 in Fig. 4 will be selected by robots for fulfilling the specified order requirements mentioned above.

The PS-ECBS algorithm operates through a two-stage iterative process. In the first stage, it relaxes path conflict constraints and optimally assigns either start points or goal points to robots by solving a MILP formulation. The second stage employs ECBS to compute collision-free paths for all robots. Whenever the objective function value in the second stage exceeds that of the first stage, the algorithm introduces additional constraints to the MILP model to resolve conflicts induced by the current assignment. This iterative refinement continues until convergence is achieved.

IV. Putting Back Task in Warehouse

A. MILP Model in Putting Back Task

In putting back task, each robot should carry one shelf at different picking stations and move toward an empty position in the corresponding storage area, as shown in Fig. 1. Similar to the picking up task, the putting back task also follows an iterative solution process. The mathematical model 2 of the $(k+1)th$ iteration is as follows:

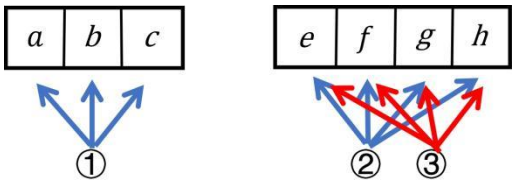


Fig. 5 Schematic diagram of the relationship between empty positions and the returning robots. ①, ② and ③ represent the robots carrying the returning shelves. The squares are the goals, where *a*, *b*, ⋯, *g*, *h* represent the number of empty positions.

$$\min \quad z \tag{11}$$

$$z \geq \sum_{j \in I} c_j x_j \tag{12}$$

$$\sum_{j \in I_{mg}} x_j = M_g \tag{13}$$

$$z - cost^k \geq (\sum_{j \in I_1^k} x_j - |I_1^k|)G \tag{14}$$

$$x_j \in \{0,1\} \tag{15}$$

Where, in the model the objective is to minimize z as shown in (11). The length c_j of the shortest path from empty position j to the corresponding picking station in (12) is the Manhattan distance, similar to c_i in (1). Set I contains all the empty positions.

Although the goals in putting back task are not preassigned, there are still some restrictions, for example, the identical commodities are usually arranged adjacent to each other. Normally only one of the restricted goals can be selected by each robot, and the number of constrained goal positions matches the returning shelves in quantity. The returning robots are divided into C groups, and the number of robots varies in different groups. M_g is the total number of the returning shelves in the gth group ($g = 1,2,3,\cdots C$, C is the maximum number of groups). If there are L_g constraints in the gth group, the set of the selected empty position for the mth constraint in the gth group could be described as I_{mg}, then we have $I = \{I_{11}, I_{12}, \cdots I_{mg} \cdots I_{LcC}\}$ and $j \in I$. The relationship between the restricted goals and returning shelves in the gth group can be described as constraint (13).

In each iteration, an additional constraint should be added to the MILP model to get closed to the convergence result. We divide the set I into two sets, one is $\{x_j^k | x_j^k = 0, j \in I_0^k\}$, and the other is $\{x_j^k | x_j^k = 1, j \in I_1^k\}$. Constraint (14) is the additional improvement to be inserted in each iteration. The variable x_j in constraint (15) indicates whether an empty position j for returning shelf is selected by the robot: if $x_j = 1$, the position is selected, otherwise $x_j = 0$.

B. Example of Putting Back Task

In putting back task, there are multiple empty positions in the storage area, the robots should select which empty positions to put shelves back into for optimal task assignment,

TABLE I
EXAMPLES OF MULTI-CONSTRAINTS OF THE ROBOTS AT DIFFERENT PICKING STATIONS ACCORDING TO THE MODES IN FIG.5 AND THE BACKGROUND IN FIG.1, m IS THE NUMBER OF CONSTRAINTS

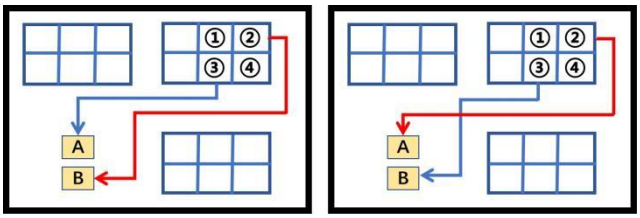
m=1 $x_{15}+x_{17}+x_{19}=1$ m=2 $x_{21}+x_{24}+x_{26}=1$ m=3 $x_{28}+x_{29}+x_{31}=2$ m=4 $x_{33}+x_{35}+x_{37}=2$	Picking Station 1
m=1 $x_{45}+x_{47}+x_{49}+x_{51}=3$ m=2 $x_{53}+x_{56}+x_{57}=2$ m=3 $x_{59}+x_{61}+x_{63}+x_{65}=3$ m=4 $x_{67}+x_{68}=1$	Picking Station 2
m=1 $x_{72}+x_{73}=1$ m=2 $x_{75}+x_{77}+x_{79}+x_{81}=3$ m=3 $x_{83}+x_{84}+x_{87}=2$ m=4 $x_{88}+x_{90}+x_{91}+x_{92}=3$	Picking Station 3

TABLE II
EXAMPLES OF MULTI-CONSTRAINTS DIVIDED INTO GROUPS BY THE DIFFERENT NUMBER OF ROBOTS

m=1 $x_{15}+x_{17}+x_{19}=1$ m=2 $x_{21}+x_{24}+x_{26}=1$ m=3 $x_{67}+x_{68}=1$ m=4 $x_{72}+x_{73}=1$	Group 1 (M_1 =1) $\sum_{j\in I_{m1}} x_j = M_1 = 1$
m=1 $x_{28}+x_{29}+x_{31}=2$ m=2 $x_{33}+x_{35}+x_{37}=2$ m=3 $x_{53}+x_{56}+x_{57}=2$ m=4 $x_{83}+x_{84}+x_{87}=2$	Group 2 (M_2 =2) $\sum_{j\in I_{m2}} x_j = M_2 = 2$
m=1 $x_{45}+x_{47}+x_{49}+x_{51}=3$ m=2 $x_{59}+x_{61}+x_{63}+x_{65}=3$ m=3 $x_{75}+x_{77}+x_{79}+x_{81}=3$ m=4 $x_{88}+x_{90}+x_{91}+x_{92}=3$	Group 3 (M_3 =3) $\sum_{j\in I_{m3}} x_j = M_3 = 3$

as shown in Fig.1. In Fig.5, for example, robot ① on the left should put one shelf back to one of the three empty positions a, b and c; and robots ② and ③ on the right should put two shelves back to two of the four empty positions e, f, g and h. We define decision variable $x_j \in \{0,1\}$, $x_j = 1$, if position j is selected by one of the robots in the same picking station, otherwise $x_j = 0$. Therefore, we have $x_a + x_b + x_c = 1$ and $x_e + x_f + x_g + x_h = 2$ respectively.

According to Fig.1, the decision variable x_j in the warehouse can be actually described as $\begin{bmatrix} x_1 & \cdots & x_{14} \\ x_{15} & \cdots & x_{28} \\ \vdots & \ddots & \vdots \\ x_{99} & \cdots & x_{112} \end{bmatrix}$, where there are three picking stations. Here we use the examples to illustrate constraint (13). We define a set of parameter settings as the examples, specified in Table I, which shows the relationship between the decision variables of empty positions and the number of returning robots with multi-constraints at three picking stations according to the modes in Fig.5 and the layout in Fig.1. As we have established the Manhaton distance in (12), we do not need to divide the constraints into groups by different picking stations. Instead, based on the number of returning robots in one group, we could divide the constraints in Table I into three groups, as shown in Table II. Table I presents different sets of endpoint position selections for each picking station, however, obtaining optimal solutions under the current format of Table I is challenging. If we restructure it into Table II, where endpoint sets with the same quantity from different picking stations are grouped together, the programming for deriving solutions will become more efficient and stable. For example, the 4th constraint in the 2nd group could be described as: $\sum_{j\in I_{42}} x_j = 2$, $I_{42} = \{83,84,87\}$, $x_{83} + x_{84} + x_{87} = 2$, it means that two distinct empty positions from the set {83, 84, 87} can be assigned to two shelf-transport robots; similarly, the 2nd constraint in the 3rd group can be described as: $\sum_{j\in I_{23}} x_j = 3$, $I_{23} = \{59,61,63,65\}$, $x_{59} + x_{61} + x_{63} + x_{65} = 3$, and so on.



(a) Robot 2 goes to B, robot 3 to A (b) Robot 2 goes to A, robot 3 to B

Fig. 6 Illustration of sorting principle for lowest-time ECBS through different paths, where robot 2 goes to the goal A or B along the red line, and robot 3 goes to the goal A or B along the blue line.

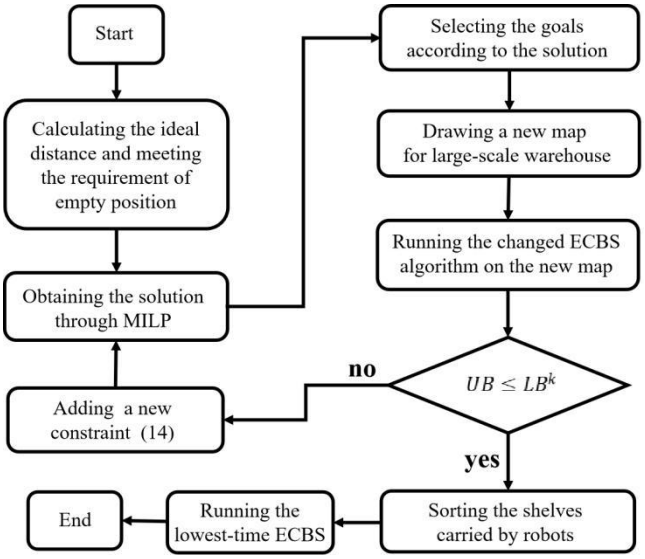


Fig. 7 Schematic diagram of PS-ECBS process for putting back task.

C. The Lowest-time PS-ECBS by Adding Sorting Principle

After the MILP solutions are obtained, the moving path of the robots at the picking station should be optimized by sorting principle to obtain the lowest time. In Fig.6, for example, the squares marked 1, 2, 3 and 4 represent the starting positions of robots at one picking station, where yellow squares denote the goals (empty positions) and there should be a little difference for the coordinates of positions 1, 2, 3 and 4.

Fig. 6 shows robot 2 and robot 3 going to the goal A or B through different paths. In Fig. 6 (a), robot 2 goes to B along the red line, robot 3 goes to A along the blue line, the total cost is 6 (blue) + 11 (red) = 17. In Fig. 6 (b) robot 2 goes to A along the red line, robot 3 goes to B along the blue line, the

TABLE III
EXAMPLE OF ITERATIVE TIMES AND RUNTIME FOR DIFFERENT ORDERS AT 32*20 IN WAREHOUSE

Samples	Order	Iterative times	Runtime(s)
Sample1	30A,26B,31C,28D	27	2.38
Sample2	25A,31B,33C,32D	83	7.17
Sample3	25A,21B,23C,22D	19	1.73
Sample4	22A,25B,23C,26D	37	3.38
Sample5	27A,23B,22C,23D	27	2.03
Sample6	21A,21B,22C,21D	8	1.16
Sample7	29A,31B,30C,32D	58	4.89
Sample8	27A,33B,32C,33D	60	5.05
Sample9	30A,32B,32C,28D	46	3.65

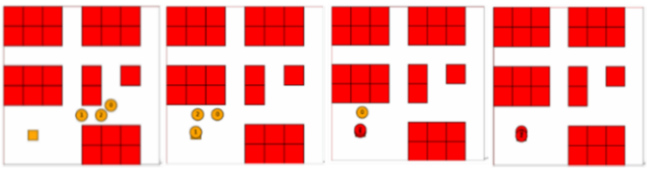


Fig.8 Video screenshot of Conflict Disappearing for three robots in one iteration at the goal. Three orange circular robots are transporting the red shelves to the orange square goal, respectively.

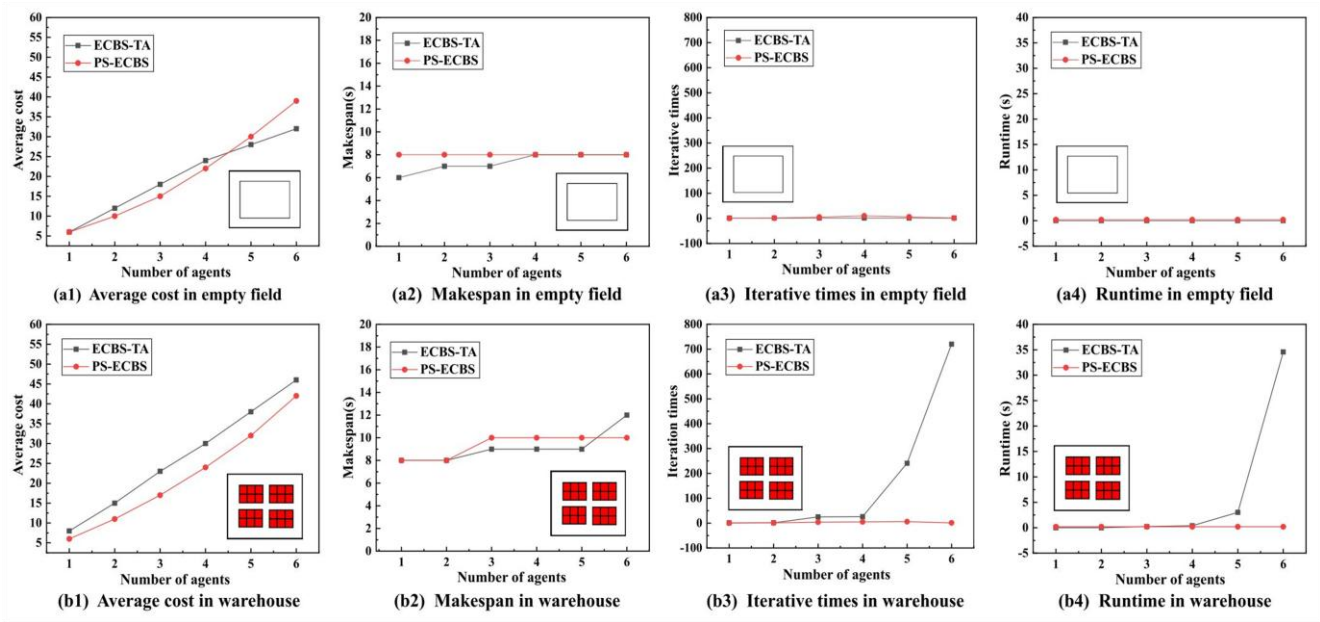


Fig. 9 Experimental results comparing PS-ECBS with ECBS-TA.

total cost is 7 (blue) + 10 (red) = 17. Although the costs in Fig. 6 (a) and Fig. 6 (b) are identical, the time spent in Fig. 6 (a) is 11, and the time spent in Fig. 6 (b) is 10, thus the latter takes less time than the former, which is related to the sorting principle. This principle requires prioritizing shelf-return sequences and assigning highest-priority shelves to most distant goal positions to obtain the lowest time. The combination of sorting principle and ECBS is called the lowest-time ECBS. The sorting algorithm will automatically eliminate a lot of unsatisfactory solutions that may take a longer time to solve. The whole process of putting shelves back through PS-ECBS is shown in Fig. 7.

V. EXPERIMENTS

A. Experiments for Picking up Tasks

Table III shows the results of iterative times and runtime under different order conditions in picking up tasks, where we used 32 × 20 grids in the experiment. It costs less than 0.2s in each iteration and the process is usually completed within a few minutes.

B. Changed ECBS by Adding Conflict Disappearing for Robots at the Goals

The collision between robots at the goal is ignored in Fig.4, but if the algorithm is put into the warehouse layout, this problem should be taken into account. To solve the problem, the original ECBS is improved by adding Conflict Disappearing for robots at the goals, resulting in changed ECBS described previously. When a robot reaches its goal, our algorithm immediately relocates it to a vacant nearby position. This creates the visual effect of "disappearance" while ensuring no more than one robot occupies any goal position simultaneously, thereby preventing goal-point collisions. Fig. 8 is a video screenshot of the moving process for three robots in one iteration generated from changed ECBS. When a robot arrives at the goal, it will disappear right now, and other robots could still reach the goal without collision.

C. Comparing the Performance of PS-ECBS in Putting Back Task with that of ECBS-TA

ECBS-TA is a typical approach that jointly optimizes task assignment and path planning in the ECBS framework [23],

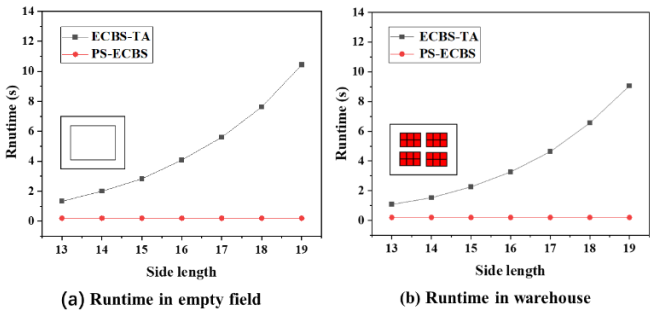


Fig. 10 Curves of runtime versus side length of the square warehouse for PS-ECBS and ECBS-TA.

whose work is similar to PS-ECBS in putting back task, where there are *M* robots at different start locations (picking stations) and *N* potential goal locations (empty positions in storage area). In comparison, PS-ECBS uses MILP model 2 to solve the multi-robot task assignment problem.

We conducted a series of preliminary experiments to compare the performance of PS-ECBS in putting back task with that of ECBS-TA, as shown in Fig. 9. In the experiments we ran both algorithms on the identical 9×7 grids in the empty field and warehouse layout with varying number of robots, but fixed group size of 6 robots per group. Fig. 9 (a1), (a2), (a3), (a4) show the experimental curves of both algorithms with the change of the number of robots in empty field, where there are slight differences in the average cost and makespan (the time elapsed until the last robot reaches its goal), but there are almost no iterative times and runtime differences between PS-ECBS and ECBS-TA.

Fig. 9 (b1), (b2), (b3), (b4) are the experimental results of both algorithms with the change of the number of robots in warehouse. In Fig. 9(b1), as the number of robots increases, the average costs for both algorithms gradually rise, with PS-ECBS having a slightly lower average cost. In Fig. 9 (b2), as the number of robots increases, the makespan of PS-ECBS is slightly higher, but with only 6 robots, the makespan of PS-ECBS is lower than that of ECBS-TA. For few robots, there are no differences in iterative times and runtime for both algorithms in Fig. 9 (b3), (b4). For more robots, there are still no obvious changes in iterative times and runtime for PS-ECBS, but the iterative times and runtime for ECBS-TA are rapidly growing with increasing of the number of robots, as shown in Fig. 9 (b3), (b4).

Based on the preliminary experiments conducted above, we expanded the warehouse area to evaluate the performances of PS-ECBS and ECBS-TA. Fig. 10 (a), (b) respectively present the runtime curve trends of both algorithms in the empty field and warehouse layout while keeping the number of shelves unchanged. The results demonstrate that as the warehouse side length increases, the runtime of ECBS-TA shows a distinct gradual upward trend, while the runtime of PS-ECBS consistently remains very short runtime without variation. This further indicates that as the warehouse side length expands, the proposed algorithm exhibits significantly superior and more stable performance compared to ECBS-TA.

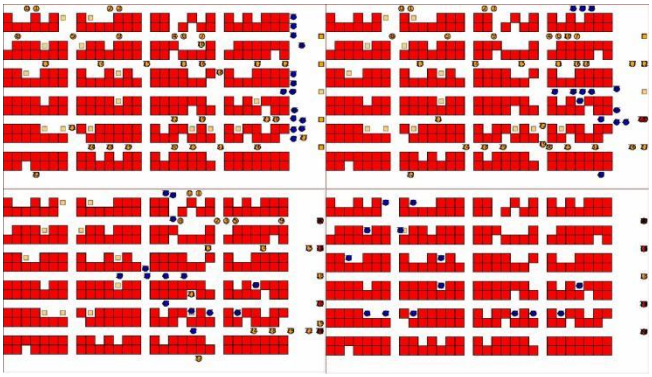


Fig. 11 Video screenshot of the logistics simulation in the warehouse at four different times, where both processes for robots to pick up and put back shelves occur simultaneously. The five yellow squares on the right side represent the goals of orange robot for picking up task, while the yellow squares between red shelves are the goals selected by blue robot for putting back task. The number of robots in work reaches 46. See the attached video.

PS-ECBS uses MILP to change the process of ergodic method, sorts the ideal distances from small to large and sequentially selects the minimum in each iteration, thus finding the best solution earlier with fewer iterations. In each iteration, an additional constraint is added to the MILP model to refine the solution selection, which enables PS-ECBS to find the convergence result as soon as possible to reduce the iterative times. Furthermore, the above lowest-time ECBS based on sorting principle can remove a lot of unsatisfactory ECBS solutions, thereby reducing runtime. Consequently, PS-ECBS runs significantly faster and has much shorter runtime compared to ECBS-TA.

D. Simultaneous Implementation of Both Picking up and Putting Back Tasks

We hope that the two processes of picking up and putting back shelves in the warehouse occur at the same time, but this would cause more collisions and increase planning complexity. In the experiment, however, this problem can be solved by the combination of Fig. 3 and Fig. 7, because they are implemented in the unified PS-ECBS framework.

In the warehouse context, the whole PS-ECBS processes were carried out, using 32 × 20 grids with the number of robots in operation ranging from 30 to 50, similar to real-world scenarios. It costs normally within a few seconds in each iteration and the whole process is usually completed within a few minutes. However, ECBS-TA would take a very long time under the same condition, making it nearly impossible to complete the actual operation. This further demonstrates that PS-ECBS could be an effective method for dealing with large-scale warehouse problems.

In the computational experiment, the best solutions selected by PS-ECBS algorithm are figured out and visualized to achieve the video of robot's movement. Fig. 11 shows the video screenshot of the logistics simulation in the warehouse at four different times, where both processes were simultaneously implemented.

VI. Conclusion

The PS-ECBS algorithm with optimal task assignment is extensively evaluated in this work, and the important results are as follows: (1) two MILP models with more constraints are formulated to dynamically select the start and end positions of the robots for optimal task assignment in the warehouse. The combination of MILP and ECBS is successfully implemented, resulting in PS-ECBS algorithm, which is a loop with multiple iterations and in each iteration an additional constraint is added to modify the model; (2) the existing ECBS is further improved by adding Conflicts Disappearing and Sorting Principle; (3) PS-ECBS runs significantly faster. It follows that the large-scale automated warehouses with multiple robots would benefit from this algorithm, especially when short runtime is required.

We believe that PE-ECBS can be used in all cases where task assignment and path planning might be optimized jointly in dense environments. In future work, we aim to combine PS-ECBS with deep learning techniques to further enhance the efficiency of multi-agent path planning in dynamic environments, particularly in warehouse logistics scenarios. Additionally, comparative studies between PS-ECBS and other established MAPF methods should be explored.

References

[1] T. Lamballais, D. Roy, and M. B. M. De Koster, "Estimating performance in robotic mobile fulfillment system," *Eur. J. Oper. Res.*, vol. 256, no. 3, pp. 976-990, 2017.

[2] L. Xie, N. Thieme, R. Krenzler, and H. Li, "Introducing split orders and optimizing operational policies in robotic mobile fulfillment systems," *Eur. J. Oper. Res.*, vol. 288, pp. 80–97, 2021.

[3] X. B. Xu and Z. Q. Ma, "Robotic mobile fulfillment systems: State-of-the-art and prospects," *Acta Autom. Sin.*, vol. 48, no. 1, pp. 1-20, 2022.

[4] H. Ma, S. Koenig, and N. Ayanian, *et al*., "Overview: generalizations of multi-agent path finding to real-world scenarios," *arXiv preprint arXiv:1702.05515*, 2017.

[5] M. N. Zafar and J. C. Mohant, "Methodology for path planning and optimization of mobile robots: A Review," *Procedia Comput. Sci.*, vol. 133, pp. 141–152, 2018.

[6] R. Stern, N. Sturtevant, and A. Felner, *et al*., "Multi-agent path finding: definitions, variants, and benchmarks," *arXiv preprint arXiv:1906.08291*, 2019.

[7] B. Li and H. Ma, "Double-deck multi-agent pickup and delivery: Multi-robot rearrangement in large-scale warehouses," *IEEE Robot. Automat. Lett.*, vol. 8, no. 6, pp. 3701–3708, Jun. 2023.

[8] M. Merschformann, L. Xie, and D. Erdmann, "Multi-agent path finding with kinematic constraints for robotic mobile fulfillment systems," *arXiv preprint arXiv:1706.09347*, 2017.

[9] N. V. Kumar and C. S. Kumar, "Development of collision-free path planning algorithm for warehouse mobile robot," *Procedia Comput. Sci.*, vol. 133, pp. 456-463, 2018.

[10] P. R. Wurman, R. D′Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no.1, pp. 9-20, 2008.

[11] H. Ma, J. Li, T. K. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," in *Proc. 16th Int. Conf. Auton. Agents Multiagent Syst.*, 2017, pp. 837-845.

[12] A. Okoso, K. Otaki, and T. Nishi, "Multi-agent path finding with priority for cooperative automated valet parking," in *Proc. IEEE Intell. Transp. Syst. Con*f., 2019, pp. 2135–2140.

[13] M. Zhong, Y. Yang, Y. Dessouky, and O. Postolache, "Multi-AGV scheduling for conflict-free path planning in automated container terminals," *Comput. Ind. Eng.*, vol. 142, pp. 106371, 2020.

[14] H. Ma, J. Yang, L. Cohen, T. K. S. Kumar, and S. Koenig, "Feasibility study: Moving nonhomogeneous teams in congested video game environments," in *Proc. Conf. Artif. Intell. Interact. Digit. Entertain.*, 2017, pp. 270–272.

[15] R. D′Andrea, "A revolution in the warehouse: A retrospective on Kiva systems and the grand challenges ahead," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 4, pp. 638-639, 2012.

[16] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numer. Math.*, vol. 1, pp. 269–271, 1959.

[17] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*," vol. 4, no. 2, pp. 100-107, 1968.

[18] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 3260–3267.

[19] H. Ma, C. A. Tovey, and G. Sharon, *et al*., "Multi-agent path finding with payload transfers and the package-exchange robot-routing problem," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 3166-3173.

[20] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent path finding," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 563-569.

[21] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015.

[22] E. Boyarski, A. Felner, R. Stern, and G. Sharon, *et al*., "ICBS: Improved conflict-based search algorithm for multi-agent path finding," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp: 740–746.

[23] M. Barer, G. Sharon, R, Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proc. Annu. Symp. Combin. Search*, 2014, pp. 19– 27.

[24] J. Li, W. Ruml, and S. Koenig, "EECBS: A bounded-suboptimal search for multi-agent path finding," *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 12353-12362.

[25] A. Andreychuk, K. Yakovlev, D. Atzmon, and R. Stern, "Multi-agent path finding (MAPF) with continuous time," in *arXiv preprint arXiv:1901.05506*, 2019.

[26] D. Atzmon, R. Stern, A. Felner, G. Wagner, R. Barták, and N. Zhou, "Robust multi-agent path finding," in *Proc. 17th Int. Conf. Auton. Agents Multiagent Syst.*, 2018, pp. 1862-1864.

[27] Z. Pan, R. Wang, Q. Bi, X. Zhang and J. Yu, "RH-ECBS: enhanced conflict-based search for MRPP with region heuristics," *Robotica*, pp.1-11, 2024.

[28] W. Hönig, S. Kiesel, and A. Tinka, "Conflict-based search with optimal task assignment," in *Proc. 17th Int. Conf. Auton. Agents Multiagent Syst.*, 2018.

[29] A. Jaitly, J. Cline, and S. Farzan, "A MILP-based solution to multi-agent motion planning and collision avoidance in constrained environments," *arXiv preprint arXiv:2506.21982*, 2025.

[30] P. Adjei and D. Masel, "RCSRS Multi-robot Path Planning Using Mixed-Integer Linear Programming (MILP): An A* Algorithm Approach," in *Intelligent Production and Industry 5.0 with Human Touch, Resilience, and Circular Economy (Lecture Notes in Production Engineering)*, D. N. Šormaz, B. Bidanda, O. Alhawari, and Z. Geng, Eds. Cham: Springer, 2025, pp. 17-26.